

RRRRRRRRRRRR	TTTTTTTTTTTT	PPPPPPPPPPPP	AAAAAAAAAA	DDDDDDDDDDDD			
RRRRRRRRRRRR	TTTTTTTTTTTT	PPPPPPPPPPPP	AAAAAAAAAA	DDDDDDDDDDDD			
RRRRRRRRRRRR	TTTTTTTTTTTT	PPPPPPPPPPPP	AAAAAAAAAA	DDDDDDDDDDDD			
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	PPP	PPP	AAA	AAA	DDD	DDD
RRRRRRRRRRRR	TTTT	PPPPPPPPPPPP	AAA	AAA	AAA	DDD	DDD
RRRRRRRRRRRR	TTTT	PPPPPPPPPPPP	AAA	AAA	AAA	DDD	DDD
RRRRRRRRRRRR	TTTT	PPPPPPPPPPPP	AAA	AAA	AAA	DDD	DDD
RRR	RRR	PPP	AAAAAAAAAAAAAAAA	DDD	DDD		
RRR	RRR	PPP	AAAAAAAAAAAAAAAA	DDD	DDD		
RRR	RRR	PPP	AAAAAAAAAAAAAAAA	DDD	DDD		
RRR	RRR	PPP	AAA	AAA	DDD	DDD	
RRR	RRR	PPP	AAA	AAA	DDD	DDD	
RRR	RRR	PPP	AAA	AAA	DDD	DDD	
RRR	RRR	PPP	AAA	AAA	DDD	DDD	
RRR	RRR	PPP	AAA	AAA	DDDDDDDDDDDD	DDDD	
RRR	RRR	PPP	AAA	AAA	DDDDDDDDDDDD	DDDD	
RRR	RRR	PPP	AAA	AAA	DDDDDDDDDDDD	DDDD	

```
DDDDDDDD  TTTTTTTTTT EEEEEEEEEEE
DDDDDDDD  TTTTTTTTTT EEEEEEEEEEE
DD      DD      TT      EE
DD      DD      TT      EE
DD      DD      TT      EE
DD      DD      TT      EE
DD      DD      TT      EEEEEEEEE
DD      DD      TT      EEEEEEEEE
DD      DD      TT      EE
DD      DD      TT      EE
DD      DD      TT      EE
DD      DD      TT      EEEEEEEEE
DD      DD      TT      EEEEEEEEE
DDDDDDDD  TT      EEEEEEEEEEE
DDDDDDDD  TT      EEEEEEEEEEE

-----
-----

DDDDDDDD  FFFFFFFFFF 000000 333333
DDDDDDDD  FFFFFFFFFF 000000 333333
DD      DD  FF      00      33      33
DD      DD  FF      00      33      33
DD      DD  FF      00      33      33
DD      DD  FF      0000      33      33
DD      DD  FF      0000      33      33
DD      DD  FFFFFFFF 00      33      33
DD      DD  FFFFFFFF 00      33      33
DD      DD  FF      00      33      33
DD      DD  FF      00      33      33
DD      DD  FF      0000      33      33
DD      DD  FF      0000      33      33
DD      DD  FF      00      33      33
DD      DD  FF      00      33      33
DDDDDDDD  000000 333333
DDDDDDDD  000000 333333

.....
.....
.....
```

```
MM      MM      AAAAAA  RRRRRRRR
MM      MM      AAAAAA  RRRRRRRR
MMMM  MMMM  AA      AA  RR      RR
MMMM  MMMM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RRRRRRRR
MM  MM  MM  AA      AA  RRRRRRRR
MM  MM  MM  AAAAAAAAAA  RR  RR
MM  MM  MM  AAAAAAAAAA  RR  RR
MM  MM  MM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
MM  MM  MM  AA      AA  RR      RR
```



.TITLE DTE\_DF03 - Sample SET HOST/DTE dialer module  
.IDENT 'V04-000'

\*\*\*\*\*  
\*  
\* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
\* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
\* ALL RIGHTS RESERVED.  
\*  
\*\*\*\*\*

\* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
\* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
\* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
\* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
\* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
\* TRANSFERRED.  
\*  
\*\*\*\*\*

\* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
\* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
\* CORPORATION.  
\*  
\*\*\*\*\*

\* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
\* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
\*  
\*\*\*\*\*

++  
FACILITY:

SET HOST/DTE

ABSTRACT:

Provide modem-specific support for autodialing on a DF03, and  
serve as example for other modem types. Activated as a sharable  
image when SET HOST ttcn: /DTE /DIAL=(number:string,MODEM\_TYPE=DF03)  
is run.

ENVIRONMENT:

VAX/VMS, user mode.

--  
AUTHOR: Jake VanNoy, CREATION DATE: 11-Apr-1984

MODIFIED BY:

\*\*\*  
  
.SBTTL DECLARATIONS  
;



; INCLUDE FILES:

; \$SHRDEF  
; \$STSDEF; shared messages  
; status fields

; MACROS:

; EQUATED SYMBOLS:

REMS\_FACILITY = ^X1FE  
REMS\_BADVALUE = SHR\$\_BADVALUE!<REMS\_FACILITY@16>  
CR = 13  
LF = 10

; OWN STORAGE:

CTRLB_DESC:	.LONG	1	; length
	.LONG	0	; will get filled in by code
CTRLB_STR:	.LONG	2	; '2' is ^B
CONN_DESC:	.LONG	CONN_STR_LEN	; length
	.LONG	0	; will get filled in by code
CONN_STR:	.ASCII	<CR><LF>/Connection made to remote port/<CR><LF>	
CONN_STR_LEN =	.-CONN_STR		
FAIL_DESC:	.LONG	FAIL_STR_LEN	; length
	.LONG	0	; will get filled in by code
FAIL_STR:	.ASCII	<CR><LF>/Failed to connect to remote port/<CR><LF>	
FAIL_STR_LEN =	.-FAIL_STR		
READ_BUFFER:	.BLKB	10	; read buffer
IOSB:	.LONG	0,0	; I/O status
READ_STATUS:	.LONG	0	; completion status
USER_CHAN:	.LONG	0	; command channel own storage

.SBTTL DTE\_DF03 - DF03 autodial routine

++

FUNCTIONAL DESCRIPTION:

Perform the necessary autodial protocol on a DF03-AC modem.

CALLING SEQUENCE:

DIAL\_ROUTINE (number\_desc, port\_chan, command\_chan)

INPUT PARAMETERS:

4(AP) - descriptor of string specified in NUMBER:string  
 8(AP) - channel number of port DF03 is connected to  
 12(AP) - channel number of user's terminal

IMPLICIT INPUTS:

NONE

OUTPUT PARAMETERS:

NONE

IMPLICIT OUTPUTS:

NONE

COMPLETION CODES:

R0 - status

SIDE EFFECTS:

NONE

--

number = 4  
 port\_chan = 8  
 command\_chan = 12

```
.TRANSFER      DIAL_ROUTINE
.MASK           DIAL_ROUTINE
BRW            DIAL_ROUTINE+2

.ENTRY DIAL_ROUTINE, ^M<R2,R3,R4>

MOVZWL command_chan(AP),user_chan      ; save for later
MOVL   number(AP),R2                   ; fetch address of descriptor
MOVZWL (R2),R3                          ; length of string
MOVL   4(R2),R4                        ; address
;
; Loop through string to check for illegal characters
;
10$: CMPB    #^A/=/, (R4)                ; '=' is pause character
    BEQL    20$                        ; branch if match
```



```

      CMPB    #^A/0/,(R4)      ; check for number
      BGTRU   30$              ; Branch if less than legal
      CMPB    #^A/9/,(R4)      ; check for number
      BLSSU   30$              ; Branch if more than legal
20$:  INCL     R4                ; next character
      SOBGTR  R3,10$           ; legal character, loop
      BRB     40$              ; continue, number ok
      ;
      ; error in number string
      ;
30$:  PUSHL   number(AP)        ; signal error
      PUSHL   #1                ; number of FA0 args
      PUSHL   #REMS_BADVALUE    ; error type
      CALLS   #3,G^CIB$SIGNAL   ; error
      MOVL    #REMS_BADVALUE!ST$SM_INHIB_MSG,R0 ; return status
      RET                                           ; return
40$:  ;
      ; number string ok, continue.
      ; queue read for character
      ;
      BSBW    READ_CHAR         ; read status character
      BLBC    R0,100$           ; exit on error
      ;
      ; Write string to modem
      ;
      MOVAB   CTRLB_STR,CTRLB_DESC+4 ; set address
      MOVAB   CTRLB_DESC,R2        ; ^B initiates dial
      BSBW    WRITE_STR          ; write string
      BLBC    R0,100$           ; exit on error
      ;
      MOVL    number(AP),R2        ; fetch address of descriptor
      BSBW    WRITE_STR           ; write number string
      BLBC    R0,100$           ; exit on error
      ;
      $HIBER_S                    ; wait for read to complete
      ;
100$: MOVL    READ_STATUS,R0      ; set status
      RET

```

.SBTTL WRITE\_STR - write string to port channel  
++

## FUNCTIONAL DESCRIPTION:

write a string to the DTE port

## CALLING SEQUENCE:

BSBW WRITE\_STR

## INPUT PARAMETERS:

R2 - address of descriptor to write

## COMPLETION CODES:

R0 - status

## WRITE\_STR:

```
$QIOW_S -  
  CHAN = port_chan(AP),-      ; channel  
  FUNC = #IOS_WRITEVBLK!IOSM_NOFORMAT,- ; write no format  
  P1   = @4(R2),-           ; address  
  P2   = (R2)               ; length  
RSB
```



.SBTTL READ\_CHAR - read status character from port

++

FUNCTIONAL DESCRIPTION:

Read the status character from the DF03, allowing a maximum of 60 seconds for the event to occur.

CALLING SEQUENCE:

BSBW READ\_CHAR

INPUT PARAMETERS:

NONE

COMPLETION CODES:

R0 - status

READ\_CHAR:

```
$QIO_S -
  CHAN = port_chan(AP),-          ; channel
  FUNC = #IOS_READVBLK!IOSM_TIMED!IOSM_PURGE,-
  -                               ; read timed, purge
  IOSB = IOSB,-                  ; I/O status
  ASTADR = READ_DONE,-           ; ast routine
  P1 = READ_BUFFER,-            ; address
  P2 = #1,-                      ; length
  P3 = #60                      ; timeout
RSB                               ; exit with status
```



.SBTTL READ\_DONE - ast for read completion

++

FUNCTIONAL DESCRIPTION:

Check for timeout or status character

CALLING SEQUENCE:

CALLED as AST routine

INPUT PARAMETERS:

NONE

COMPLETION CODES:

R0 - status

--

.ENTRY READ\_DONE,^M<R2>

MOVZWL IOSB,R0 ; get status of read  
BLBC R0,100\$ ; branch if timeout

MOVZBL READ\_BUFFER,R2 ; fetch data  
CMPB #^A/^/,R2 ; status ok?  
BNEQ 10\$ ; branch if not  
MOVAB conn\_desc,R2 ; set up string  
MOVAB conn\_str,4(R2) ; set up string  
BSBW WRITE\_STR\_TO\_USER ; tell user, ready  
BLBC R0,100\$ ; exit on error  
MOVL #SS\$\_NORMAL,R0 ; ready  
BRB 100\$ ; exit

10\$:

MOVAB fail\_desc,R2 ; set up string  
MOVAB fail\_str,4(R2) ; set up string  
BSBW WRITE\_STR\_TO\_USER ; tell user, ready  
MOVZWL #SS\$\_RANGOP,R0 ; status

100\$:

MOVL R0,READ\_STATUS ; save status  
\$WAKE\_S ; wake main stream

RET

.SBTTL WRITE\_STR\_TO\_USER - write string to command channel

++

FUNCTIONAL DESCRIPTION:

write a string to the user terminal channel

CALLING SEQUENCE:

BSBW WRITE\_STR\_TO\_USER

INPUT PARAMETERS:

R2 - address of descriptor to write

COMPLETION CODES:

R0 - status

--

WRITE\_STR\_TO\_USER:

\$QIOW_S -	
CHAN = user_chan,-	; channel
FUNC = #IOS_WRITEVBLK,-	; write
P1 = @4(R2),-	; address
P2 = (R2)	; length







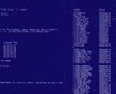















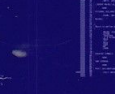











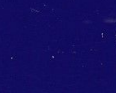



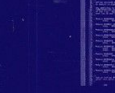
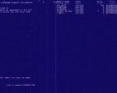










































































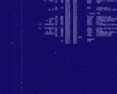




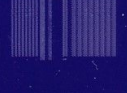






















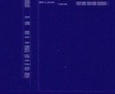
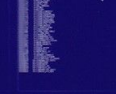



















































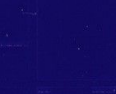
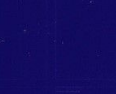










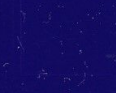




RSB

.END



0332 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

	RPGMSGTX LIS																
						DTE_DF03 MAP											
RPGMOVE3 LIS																	
			RPGSORT LIS														
	RPGOPEN LIS					RTPAD											
						CTDRIVER MAP											
																	
																	
						RTPAD MAP					RTPADMACS MAR						
RPGMSGPTR LIS																	
																	
						RPGVECTOR LIS											
	RPGPRINT LIS		RPGUPDATE LIS					RTDEF SDL		DTE_DF03 MAR	CTDRIVER LIS	